Los Alamos NATIONAL LABORATORY EST.1943

LA-UR-15-27806 (Accepted Manuscript)

Analyzing the Effectiveness of a Frame-Level Redundancy Scrubbing Technique for SRAM-based FPGAs

Tonfat, Jorge
Kastensmidt, Fernanda Lima
Rech, Paolo
Reis, Ricardo
Quinn, Heather Marie

Provided by the author(s) and the Los Alamos National Laboratory (2016-11-23).

To be published in: IEEE Transactions on Nuclear Science

DOI to publisher's version: 10.1109/TNS.2015.2489601

Permalink to record: http://permalink.lanl.gov/object/view?what=info:lanl-repo/lareport/LA-UR-15-27806

Disclaimer:
Approved for public release. Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

RESEARCH LIBRARY

Analyzing the Effectiveness of a Frame-level Redundancy Scrubbing Technique for SRAM-based FPGAs

Jorge Tonfat, *Member, IEEE,* Fernanda Lima Kastensmidt, *Member, IEEE*, Paolo Rech, *Member, IEEE,* Ricardo Reis, *Senior Member, IEEE,* and Heather M. Quinn, *Senior Member, IEEE*

***Abstract*—Radiation effects such as soft errors are the major threat to the reliability of SRAM-based FPGAs. This work analyzes the effectiveness in correcting soft errors of a novel scrubbing technique using internal frame redundancy called Frame-level Redundancy Scrubbing (FLR-scrubbing). This correction technique can be implemented in a coarse grain TMR design. The FLR-scrubbing technique was implemented on a mid-size Xilinx Virtex-5 FPGA device used as a case study. The FLR-scrubbing technique was tested under neutron radiation and fault injection. Implementation results demonstrated minimum area and energy consumption overhead when compared to other techniques. The time to repair the fault is also improved by using the Internal Configuration Access Port (ICAP). Neutron radiation test results demonstrated that the proposed technique is suitable for correcting accumulated SEUs and MBUs.**

***Index Terms*— Fault tolerance, field programmable gate arrays, radiation effects.**

I. Introduction

SRAM-based FPGAs are attractive to critical applications due to their high performance, reduced power consumption, and reconfiguration capability. However, FPGAs are highly susceptible to radiation effects such as soft errors in the SRAM configuration memory. These soft errors have a persistent effect and will remain until the restoration of the original content.

The configuration memory of SRAM-based FPGAs is arranged into small segments named *"configuration frames"*, and it represents the largest portion of all the memory cells in the device. Other factors that increase the susceptibility to soft errors are the reduction of the transistor size and the lower voltage operations of these SRAM memory cells [1].

When an ionizing particle flips a single memory cell, the event is called *Single Event Upset (SEU).* Moreover, when it impacts multiple memory cells due to charge sharing, the event is called *Multiple Cell Upset (MCU).* If the multiple corrupted bits belongs to the same memory word (or frame for FPGAs), the MCU is called *Multiple Bit Upset (MBU).* In the first families of SRAM-based FPGAs, MBU events represented a smaller percentage of total *Single Event Effects (SEEs)* [2] [3]. However, in recent technology nodes (65nm and beyond) the percentage of MBU events is higher [3]. Additionally, the trend is to reduce the technology node in future devices, which will increase the MBUs probability of occurrence. MBU events can be up to 10% of the total bit upsets observed in the configuration memory bits of an FPGA fabricated in 28nm technology [4].

An effective method to correct bit upsets in the configuration memory is memory scrubbing. The original configuration memory is restored using the dynamic reconfiguration feature of FPGAs. Different scrubbing methodologies have been proposed in the literature for Xilinx FPGAs [5] based on Error Correction Codes (ECC) or configuration data redundancy. In this work, we use a novel scrubbing technique (FLR-scrubbing) that can be implemented in a coarse grain TMR design. It is based on having each redundant TMR domain with the same configuration frame information to allow vote out the frames copies to restore themselves. This method achieves low area overhead and low energy consumption, also a reduced time to repair a bit upset when compared to other works in the literature [6], which are based on ECC or external reference (*golden*) memory.

The effectiveness of this novel scrubbing technique to correct multiple accumulated bit upsets in the configuration memory is analyzed by fault injection and radiation experiments. The FLR-scrubbing technique was implemented on a mid-size Xilinx Virtex-5 FPGA device used as a case study.

The rest of the paper is organized as follows: Section II presents selected related works. Section III describes the FLR-scrubbing technique. Implementation results and comparisons with other similar works are presented in Section IV. In Section V are given the characteristics of the methodology to evaluate the effectiveness of the FLR-scrubbing. The analysis of the neutron radiation results is provided in Section VI. A complementary fault injection campaign is presented in

Manuscript received XXX; revised XXX; accepted XXX. This work was supported in part by CNPq, FAPERGS and CAPES.

J. Tonfat, F. L. Kastensmidt, P. Rech and Ricardo Reis are with PGMICRO, UFRGS, Porto Alegre 91501-970, Brazil (e-mail: jorgetonfat@ieee.org; fglima@inf.ufrgs.br; prech@inf.ufrgs.br; reis@inf.ufrgs.br).

H. Quinn is with Los Alamos National Laboratory, Los Alamos, NM 87545 USA (e-mail: hquinn@lanl.gov).

Section VII with the objective of finding the maximum number of accumulated bit upsets the technique can correct in a determined area. Finally, conclusions and future work are discussed in Section VIII.

II. Related Work

Fault masking techniques such as Triple Modular Redundancy (TMR) are used to improve the radiation tolerance of circuits implemented in SRAM-based FPGAs. Still, it is necessary to avoid bit upset accumulation in the configuration memory with a correction mechanism to increase the reliability of the circuit [7].

Memory scrubbing is a well-known correction method for the configuration memory of SRAM-based FPGAs [5] and it can be implemented with different architectures and methodologies.

The circuit that implements the memory scrubbing is commonly known as a *scrubber*, and it can be implemented externally or internally to the FPGA as shown in [8]. An external scrubber uses external configuration ports, such as the SelectMAP (Xilinx) or JTAG port, to access the configuration memory.

On the other hand, internal scrubbers use the ICAP (Internal Configuration Access Port) block [9] in Xilinx FPGAs. This port has the same interface as the SelectMAP port, but it can be accessed from the internal configurable logic. External scrubbers are presumed to be more radiation tolerant than internal scrubbers since they can be implemented as a rad-hard ASIC or in a rad-hard antifuse-based FPGAs. In [8] is presented a comparison of the reliability between an external and an internal scrubber. Results show that an external scrubber has an improved correction capability (corrected all SEUs in the configuration memory) and does not present any SEFI that required a power cycle in contrast to the results obtained from the internal scrubber. However, it is possible to apply hardened by design techniques to internal scrubbers to improve its reliability as shown in [10] and [11].

Fig. 1 depicts two possible scenarios when multiple bit upsets occur in configuration frames. When an impinging particle corrupts cells belonging to different frames, provoking at most one bit upset per frame, the event is called an inter-frame MBU. When an impinging particle corrupts cells belonging to the same frame, provoking multiple bit upsets, this event is called intra-frame MBU. This difference is important when Error Correction Codes (ECC) techniques are applied to repair frames, as the current ECCs usually correct single bit upsets and detect at most two upsets per frame.

Since Xilinx Virtex-4 FPGAs, all FPGAs come with a self-correction mechanism that uses Single Error Correction Double Error Detection (SECDED) ECC and Cyclic Redundancy Check (CRC). SECDED is implemented in each configuration frame, and CRC is calculated for the whole configuration memory. It is calculated the ECC syndrome of each frame to find the bit upset position and then correct it. It is the fastest method to correct a fault because it only needs to read and write one frame. The main drawback of the technique is its limited correction capability. Only one upset per frame

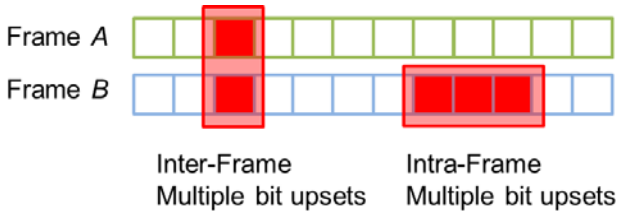


Fig. 1. Examples of Inter-Frame and Intra-Frame Multiple bit upsets.

can be corrected and at most two upsets per frame can be detected. In other words, SECDED ECC will correct inter-frame MBUs but only detect intra-frame MBUs. Xilinx offers a soft-core named Xilinx SEU Controller [16] to manage the self-correction mechanism of Virtex-5 FPGAs.

In this work, the proposed scrubbing technique can correct any multiple bit upset pattern, with equivalent upset repair time of the Xilinx self-correction mechanism based on ECC.

More recently, Xilinx offered a new soft-core named Soft Error Mitigation (SEM) Controller for the 7-series family [12] that also uses the ICAP interface. This core improves the capability of the previous one by combining the SECDED and the CRC features. Now it can correct up to two adjacent bit upsets in one frame. Moreover, this core offers the possibility to correct the configuration memory by replacing it with a *golden* external reference. This reference is a copy of the configuration memory, usually stored in an external rad-hard memory.

This core can correct intra-frame MBUs with any number of bit upsets by using an external memory reference. However, using this memory in each scrub cycle, increments the power consumption of the system; and the upset repair time is limited by the reading access time of this external memory. Usually, the external memory is slower than the FPGA configuration memory. For example in our case study, the ICAP port needs 98 clock cycles running at a maximum frequency of 100 MHz to read one frame. Instead, it is required 574 cycles running at 50 MHz to read one frame from an external flash memory. This approach increments the Mean Time to Repair (MTTR).

Some scrubbing techniques have been proposed to avoid the use of an external memory and maintain the capability to correct intra-frames MBUs. In [13] is presented a self-reference inter-FPGA scrubber, so no external memory is used. In this case is applied a coarse grain TMR at device-level, so at least three identical FPGAs are needed to implement this technique. Each FPGA has the same configuration data. Consequently, the power consumption is incremented at least three times. This technique also requires another device to hold the inter-FPGA scrubber logic and the output majority voter of the outputs of each FPGA.

In [14], it is presented another approach to protect frames against MBUs. An interleaved 2-D parity scheme is used to detect upsets in the configuration memory. These extra parity bits are stored in BRAM blocks. In this work, the detection capability is conditioned to the MBU pattern. Moreover, the correction scheme is based on a type of ECC named Erasure Codes. In this scheme, the configuration frames are grouped in

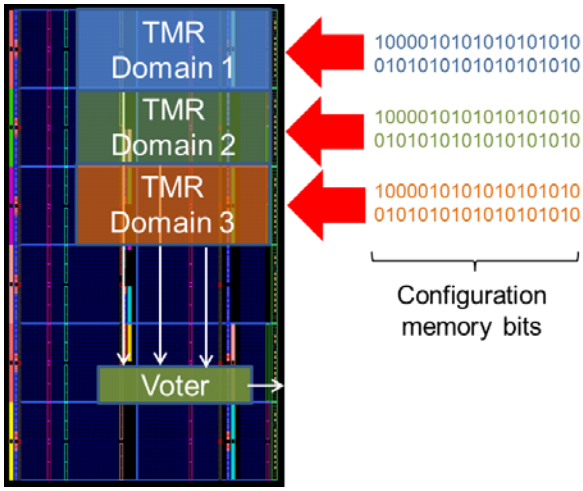


Fig. 2. Block diagram of the frame redundancy scheme based on a coarse grain TMR design.

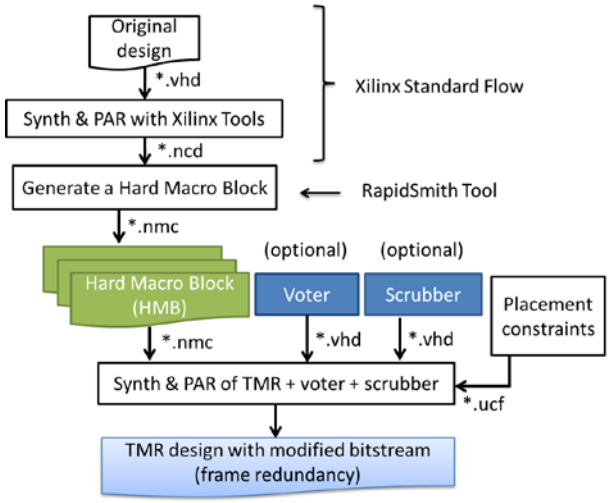


Fig. 3. Customized design flow.

clusters, and for each cluster, a redundant frame with parity information is needed. So, once a faulty frame is detected, the scrubber needs to read all the frames in the cluster of the faulty frame to reconstruct the original information. Consequently, the upset repair latency depends on the size of the cluster.

III. Proposed Frame-Level Redundancy Scrubbing Technique (FLR-Scrubbing)

The proposed scrubbing technique depends on a coarse grain TMR design, where each TMR domain has the same frame data. So, each configuration frame of the TMR design is triplicated in the FPGA. Therefore, any frame of the TMR design can be repaired using the information of the other two identical frames. In this proposed technique, coarse grain TMR is used to mask faults at the circuit level and also to enable the correction of the faults in the configuration memory.

The description of the FLR-scrubbing technique is divided into two parts:

- In the first part, the process to generate the coarse grain TMR circuit with a customized design flow is detailed. This process ensures that the configuration frames in each TMR domain are the same.
- In the second part, a new scrubbing logic that uses the information of the triplicated configuration frames to correct bit upsets is described.

Fig. 2 shows a block diagram of a coarse grain TMR design implemented using this technique.

A. Customized Design Flow

The TMR design needs a particular placement to obtain the same configuration frames for each TMR domain. Also, each TMR domain should be implemented in the FPGA with same resources and routing.

The placement of each TMR domain is obtained by placement constraints, and it depends on the structure of the configuration memory of the target FPGA. In Xilinx FPGAs, the configuration memory is structured as an array of configuration frames vertically placed in the matrix. One frame fits into a single row, and each column may have many vertical frames. For example in a Xilinx Virtex-5 FPGA, a CLB row and column has 36 frames each one with 1312 bits disposed vertically. Consequently, the height of a frame has 1312 bits, and each row has the height of a frame. TMR domains must be placed vertically aligned, and covering an integer number of rows, as shown in Fig. 2.

This technique cannot be applied to the majority voter due to the limitations of triplicating the majority voter with three identical placements of logic and inputs. So, there are two possible solutions to correct upsets in the voters. The first one is to have a copy of the majority voter configuration frames in a memory and load this frames when needed to restore the corrupted frames of the majority voter. The second solution is to place the voter outside the FPGA. This option implies the use of another device as a rad-hard antifuse FPGA; however, it may increase the system complexity and board area, but assuring the voter functionality under radiation effects.

To obtain a coarse grain TMR design, where each TMR domain is synthesized, placed and routed in the same manner, it is proposed a customized design flow as shown in Fig. 3. This design flow is partially based on the Xilinx Standard design flow that uses the commercial Xilinx tools to generate an FPGA bitstream from a hardware description of the design; and it is also based on the RapidSmith tool [15], an academic tool to generate Hard Macro Blocks (HMB) that assures that each TMR domain has the same configuration frames.

These HMB are used to implement each of the TMR domains because these blocks are already placed and routed designs that can be instantiated in an FPGA design.

The proposed design flow begins with the generation of a placed and routed circuit from the original design with the Xilinx standard design flow. Then, the RapidSmith tool receives the placed and routed circuit in an NCD format and creates the HMB with an NMC format.

The second step is to create a new design project, which includes the three instances of the HMB with the placement constraints. The voter of the coarse grain TMR design and the scrubber circuit can be optionally included in the project because they can be implemented outside the FPGA. Xilinx

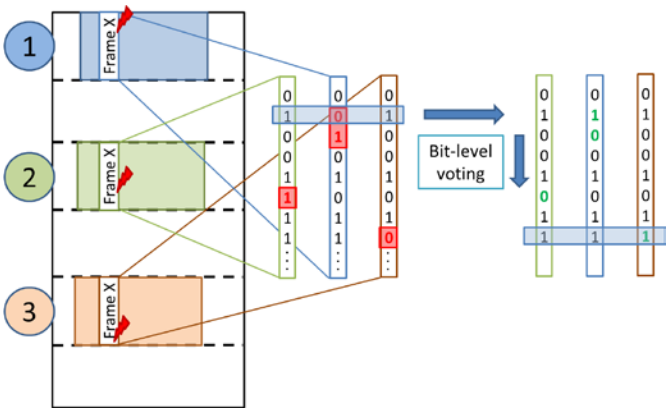


Fig. 4. Procedure to correct three identical frames with bit-level majority voting.

tools support the combination of HDL designs with HMB to generate the final bitstream.

B. *The Scrubbing Technique*

The customized design flow ensures that the three TMR domains have the same configuration frames. The FLR-scrubbing technique uses this information to correct these regions.

The scrubber circuit needs to be configured with the location of the three TMR domains. When the scrubber begins a scrub cycle, it reads the first frame of each TMR domain. Then, the scrubber executes a bit level majority voting of the three frames and creates a fault-free (voted) frame. In the best-case scenario, none of the frames needs to be corrected, so the scrubber moves to the next frame position of each TMR domain. If any of the three frames is corrupted, the scrubber replaces it with the fault-free (voted) frame.

In the worst-case scenario, three frames are corrupted due to the accumulation of bit upsets. The scrubbing technique will be able to correct all the upsets if there is no more than one upset per each relative bit position, as shown in Fig. 4. Please note that the frame from the TMR domain 1 has an MBU, which would not be corrected by ECC [16], but because our method votes bit by bit, the MBU of the frame from TMR domain 1 will be voted out correctly.

The correct execution of the scrubbing is based on the assumption that at most one of the three bits voted is faulty. This scenario is expected because the three bits compared by the voter are not physically adjacent, so it is very unlikely to have multiple bit upsets in two of the three bits. We have not observed such cases under radiation to an accumulation up to 274 bit upsets in average. However, if such unlikely case occurs, the scrubber will not detect nor correct the frames and golden bitstream must be loaded from an external memory.

IV. Implementation Results

The FLR-Scrubbing technique is implemented into a Xilinx Virtex-5 FPGA, part XC5VLX50T-FFG1136. This device is manufactured with a 65 nm technology, and it has a nominal core voltage of 1.0V. It is worth noting that although, in this study we consider Xilinx FPGAs, the proposed technique is generic and extendable to any SRAM-based FPGA that offers

TABLE I
Area Comparison Results

Scrubbing Scheme	CLBs	BRAMs	External Memory
Work in [14] [a]	1100 (6%)	4 (1%)	No
Xilinx SEU Controller [16]	98 (3%)	1 (2%)	No
Xilinx SEM Controller [12] [b]	108 (2%)	3 (1%)	Yes
Blind Scrubbing (TMR)	341 (10%)	12 (20%)	Yes
FLR-SCRUBBING (TMR)	**113 (3%)**	**6 (10%)**	**No**

[a] implemented for a Virtex-6 VLX240T FPGA with 50 clusters and TMR redundancy. In this device, one frame has 2,592 bits.

[b] implemented for an Artix-7 A100T FPGA without optional features. In this device, one frame has 3,232 bits.

configuration memory readback. The scrubber circuit is within the FPGA, and it uses the ICAP block to access the configuration memory. This block can work at a maximum frequency of 100MHz with a 32-bit data interface.

A. *Area Overhead*

The area overhead of the FLR-scrubbing technique is low, and it corresponds to the area of the scrubber circuit because it does not need any extra memory to store frame parity bits or copies of the original frames. When comparing to techniques such as [14] that need extra memory to store parity bits or [8] [12] that needs external memory to read the original (*golden*) configuration memory, our method can show a good advantage. The area overhead of the FLR-scrubbing technique is also independent of the size of the FPGA, and it is similar to Xilinx SEU Controller [16]. The area of Xilinx SEM Controller [12] depends on the size of the FPGA and the selected error correction method that can be based on the embedded ECC bits only, a combination of ECC and CRC or the external *golden* memory.

Related works compared in this paper [12],[14],[16] do not mention any mitigation technique for the scrubber circuit. In our method, we propose to triplicate the scrubber to improve the reliability. The presented area considers the triplication of the scrubber circuit. Table I presents our results compared with previous works.

B. *Time to Repair One Configuration Frame*

The time to repair one frame is the time needed to correct one or more bit upsets in one configuration frame. As mentioned in section III, in the proposed scheme, not all the configuration frames are scanned. Only the configuration frames of the TMR circuit are analyzed. These frames contain the potential bits that, if corrupted, can generate an error in the circuit. In our case, the time to repair one frame depends on the time needed to read the three selected frames and the time to write the fault-free (voted) frame.

The work in [14] needs to read a cluster of configuration frames to correct one frame. Reducing the number of frames per cluster, reduces the time to repair one frame but increments the area overhead.

TABLE III
COMPARISON OF ENERGY AND POWER CONSUMPTION FOR THE FLR-SCRUBBING TECHNIQUE AND BLIND SCRUBBING

	FLR-scrubbing (TMR)	**Blind Scrubbing (TMR)**		
Parameter	**FPGA Core Source (1.0 V)**	**FPGA Core Source (1.0 V)**	**Flash Memory Source (1.8 V)**	**All Sources**
Dynamic Power (RMS)	33.24 mW	21.2 mW	3.93 mW	25.13 mW
Scrub cycle time	3.67 ms @ 50MHz	178.55 ms @ 50MHz		
Energy per scrub cycle	121.9 µJ	3.785 mJ	0.7 mJ	4.485 mJ
Number of protected frames	1386	8376 (all the device)		
Energy to scrub a frame ($E_{scrub\text{-}frame}$)	**87.9 nJ**	451.9 nJ (84.3%)	83.89 nJ (15.7 %)	**535.79 nJ**

TABLE II
COMPARISON OF THE TIME TO REPAIR ONE FRAME

Scrubbing Scheme	**Characteristics**	**Time to repair one frame (ms)**
Work in [14] [a]	Uses BRAM to store parity frame bits	0.351
Xilinx SEU Controller [16]	The scrubber is based on a PicoBlaze	0.240
Xilinx SEM Controller [12] [b]	Uses an external memory (replace method)	0.012
Blind Scrubbing (TMR)	Uses an external memory	0.021
FLR-SCRUBBING (TMR)	**BRAM or external memory is not needed**	**0.005**

[a] implemented for a Virtex-6 VLX240T FPGA with 50 clusters and TMR redundancy. In this device, one frame has 2,592 bits.

[b] implemented for an Artix-7 A100T FPGA without optional features. In this device, one frame has 3,232 bits.

The Xilinx SEU Controller [16] should be the fastest approach because the procedure to repair one frame needs only to read the corrupted frame and use the internal ECC bits to correct one single bit upset per frame. However, it is implemented with an 8-bit PicoBlaze microcontroller, so the performance of the scrubber is reduced.

The time to access the external memory is slow compared to the time to access the internal frame using the ICAP. As shown in [17], the relation between both access times can be nearly 100 times. Consequently, the time to repair by using Xilinx SEM controller [12] is much higher than the FLR-scrubbing technique as it uses the external memory access.

The results between the proposed method and other approaches are summarized in Table II.

C. Energy Consumption

We cannot neglect the power overhead imposed by the configuration memory scrubbing. The power consumption of the scrubbing process depends on the readback or scrub rate and the scrub methodology adopted. So, the energy consumed per configuration frame ($E_{scrub\text{-}frame}$) was used as a parameter to compare different scrubbing methods. This parameter will give us a better idea of the energy efficiency of the technique and is independent of the scrub rate or the scrub methodology. A detailed description of the process to measure this parameter is found in [6].

We compare the energy efficiency of the FLR-scrubbing technique against a blind scrubbing methodology. We decide to choose the blind scrubbing methodology as a reference point since it implements the simplest scrubbing methodology. The blind scrubber implemented uses an external flash memory with a 16-bit parallel interface.

The energy consumed by the FLR-scrubbing technique comes only from the FPGA core. On the other hand, the blind scrubbing methodology consumes power from the FPGA core and the external flash memory.

The results are summarized in Table III. It is observed that the energy consumption to scrub one frame of the FLR-scrubbing is at least six times less than the blind scrubbing. The energy consumption does not depend on the scrubbing rate or the methodology.

In the comparison, the blind scrubber covers all the device configuration frames. On the other hand, the FLR-scrubber is only correcting a specific area of the FPGA where the functional design is placed. It is impossible to cover the whole FPGA because the location of the ICAP block and the internal scrubber logic prohibits the implementation of the proposed customized design flow explained in Section III.A. This issue can be solved by implementing the scrubber outside of the FPGA.

V. EXPERIMENTAL METHODOLOGY

A. Device Under Test (DUT) and Neutron beam

The FLR-Scrubbing technique was tested into the same Xilinx Virtex-5 FPGA, part XC5VLX50T-FFG1136. Experiments were performed at Los Alamos National Laboratory (LANL), Los Alamos Neutron Science Center (LANSCE) Irradiation of Chips and Electronics House II in December 2014. The FPGA device was tested at normal incidence with an approximated neutron flux of 1.43×10^6 (n/cm^2×s). The neutron energy spectrum resembles the atmospheric one between 1 and 750 MeV [18].

B. Radiation test setup

The board was placed in the radiation chamber and was connected to a host computer via two USB connections. The first connection is used for the FPGA configuration and readback via JTAG while the second connection is used for the RS232-C communication with the scrubber.

The objective of the test is to analyze the efficiency of the proposed scrubbing methodology to correct multiple

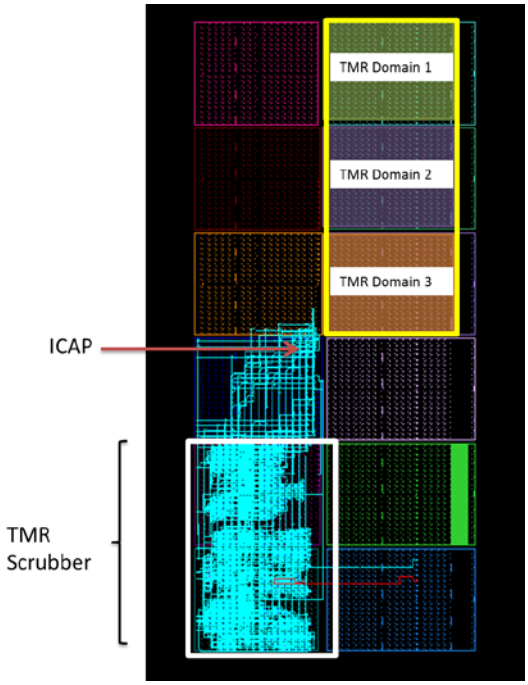


Fig. 5. Placement of the TMR scrubber and the three TMR domain zones on a Virtex-5 VLX50T.

accumulated SEUs and MBUs. Therefore, the scrubber is protected with TMR.

Fig. 5 shows the placement of the scrubber, and the three TMR domain zones that the FLR-scrubbing technique protects. The scrubber protects an area of 1,386 configuration frames, where the TMR design is located. This area represents 720 CLBs (20 % of device CLBs) and 24 18K BRAM blocks (20 % of device BRAM blocks).

The test procedure starts with the configuration of the FPGA, including the scrubber circuit and the TMR protected zone. The FPGA is exposed to radiation and to the accumulation of bit upsets in the configuration memory. The FLR-scrubbing technique is configured with two pre-defined scrubbing rate during the test: one of 30 minutes and other one of 60 minutes. During the accumulation period, periodic readbacks are performed with intervals of 5 minutes to analyze the accumulation of upsets. So, once 30 minutes or 60 minutes are reached, the FLR-scrubbing technique is activated to correct the TMR protected zone.

In order to save every scrubbing execution in a log file, when the scrubber finds bit upsets in the TMR protected zone, it transmits to the host computer the frame address of corrupted frames. The host PC executes a last readback to verify if TMR protected zone was correctly scrubbed and if there is no remaining bit upsets. The host PC reconfigures the FPGA with the original bitstream and a new run begins.

VI. Neutron Radiation Results

Experimental results are classified in four cases:

- Case 1: The scrubber sends a correct report to the host PC, and the readback files confirm the right operation of the scrubber.
- Case 2: The scrubber sends a wrong response to the host PC, and the readback files confirm an error on the scrubber.
- Case 3: The scrubber sends a correct report to the host PC, but the readback files present a wrong scrubbing correction.
- Case 4: The scrubber sends a wrong response to the host PC, but the readback files present a correct scrubbing operation.

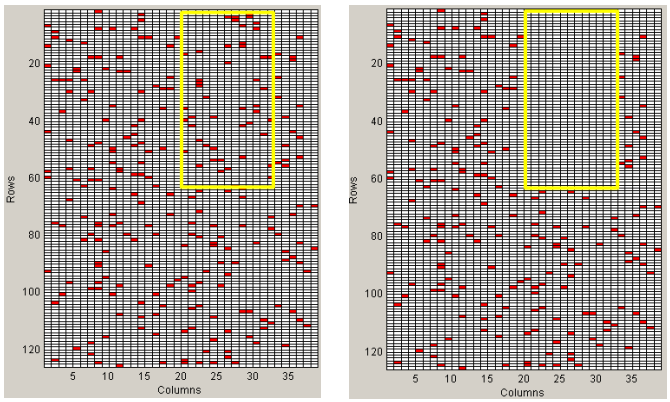


Fig. 6. Graphical view of readbacks of a correct run (case 1). The spots are bit upsets in the configuration memory. (a) Readback just before the scrubber is activated. (b) Readback after the scrubber finished.

TABLE IV
Classification of Results

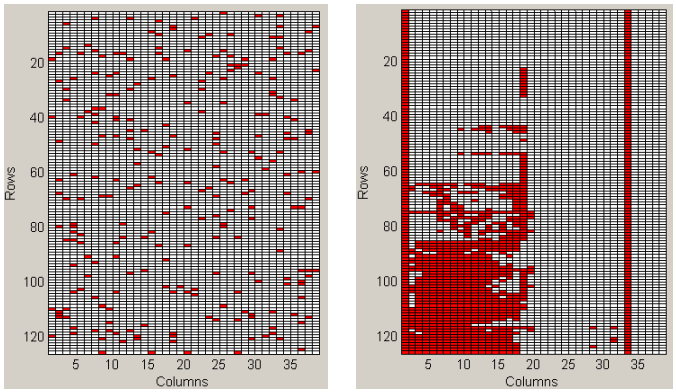
	Avg. acc. upsets	Total runs	Case 1	Case 2	Case 3	Case 4
Scrub rate 1 (30 min)	145.7	33	87.8%	6.1%	6.1%	0%
Scrub rate 2 (60 min)	274	30	66.7%	26.7%	3.3%	3.3%

The percentage of occurrence of the four cases (1 to 4) is shown in Table IV. One can see the average number of accumulated upsets per radiation test runs, the number of radiation test runs and the percentage of occurrence of each one of the four cases described above.

Case 1 represents the correct behavior of the scrubber. In this case, the TMR in the scrubber was adequate to mask any functional error, and the proposed technique was able to correct all the accumulated bit-flips in the protected area. A graphical view of readback files before and after the scrubbing is applied is shown in Fig. 6.

Cases 2, 3 and 4 are wrong behaviors of the scrubber. Cases 2 and 4 are detectable errors since the response of the scrubber shows a functional error. It is not possible to diagnose if the functional error is in the scrubber circuit or the internal configuration controller (ICAP) of the FPGA. Fig. 7 shows a representative example of a wrong behavior.

Case 3 is an undetectable functional error that was only observed after the analysis of the readback files. This case is the candidate case for finding an example where the scrubbing technique is not able to correct or detect a fault when, as described in section III.B, more than one bit of the same



(a) (b)

Fig. 7. Graphical view of readbacks of an incorrect run (cases 2 and 3). (a) Readback just before the scrubber is activated. (b) Readback after the scrubber finished.

TABLE V
CROSS-SECTION AND FAILURE IN TIME AT NYC

	cross-section (cm^2)	FIT
Scrub rate 1 (30 min)	4.41E-11	5.73
Scrub rate 2 (60 min)	6.59E-11	8.57
Xilinx SEU Controller [16]	N/A	8.6

relative bits position of the voted frames has an upset. However, after the analysis of the readback files, it was not found any evidence of such scenario. All the runs, when the scrubber failed to correct the configuration memory of the protected area, were due to a functional error in the scrubber itself or because of a Single Event Functional Interrupt (SEFI) in the ICAP block. So, this means that the error was in the scrubber and not in the technique itself.

Table V presents the cross-section and Failures in Time (FIT) results of the scrubber circuit considering the three cases of functional errors observed during the tests. As expected, the reliability of the scrubber is reduced when more bit-flips are accumulated. The FIT is obtained considering a flux of 13 neutrons /(cm^2×h) (The New York City reference flux [19]). As a reference, it is also mentioned the estimated soft error rate of the Xilinx SEU Controller [16]. Please note that in this radiation test the focus is to demonstrate the effectiveness of the FLR-scrubbing technique and not the reliability of the scrubber. Nevertheless, the scrubber reliability is critical for a real application.

VII. FAULT INJECTION ANALYSIS

The analysis of the results of the neutron radiation experiment showed that the FLR-scrubbing technique was not defeated due to accumulated bit upsets in the protected area. The objective of the fault injection campaign is to determine the maximum number of accumulated bit upsets that the scrubbing mechanism can correct.

The fault injector used in this fault injection test is described in [20]. The fault injector is also implemented in the FPGA and shares the ICAP block with the scrubber circuit. The emulated upsets are a combination of random generated

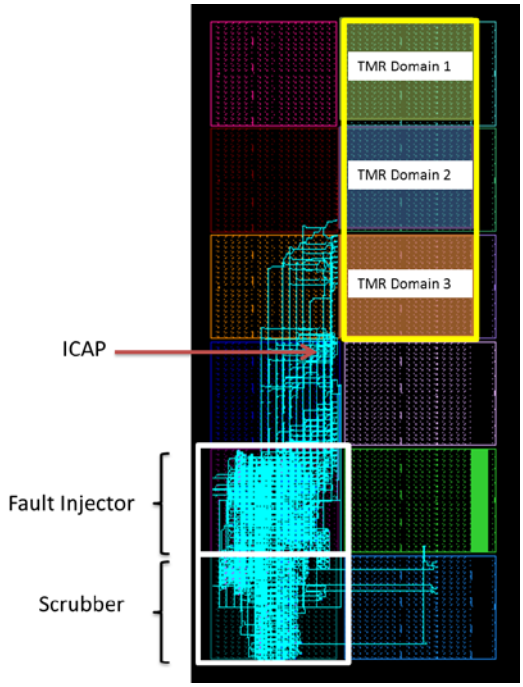


Fig. 8. Placement of the scrubber, fault injector and the three TMR domain zones on a Virtex-5 VLX50T.

TABLE VI
FAULT INJECTION RESULTS

Average maximum accumulated SEUs corrected	1136.6 (0.06% of total bits in the evaluated area)
Standard deviation (σ)	562.5

SEUs locations and locations recorded from previously accelerated neutron radiation tests. The scrubber protects the same area mentioned in the neutron radiation test (1,386 frames), and the fault injector is constrained to inject faults in the same area protected by the scrubbing technique.

Fig. 8 shows the final placement of the scrubber, the fault injector and the three identical regions where the TMR design is located.

The methodology to find the maximum number of accumulated SEUs was to inject faults in steps of 10 faults per injection and then activate the scrubber to analyze if all the faults were corrected. When faults remain after executing the scrubbing mechanism, the fault injection campaign is ended.

One hundred fault injection campaigns were performed, and more than 113,600 faults were injected. The results are summarized in Table VI.

The results show that the technique can correct more than one thousand faults on average in the protected area. This value represents a small percentage of the total configuration bits in this area, but it represents a high accumulation time of faults in a real environment.

The maximum number of accumulated bit upsets obtained in the fault injection campaign is ten times higher than the bit upsets accumulated during the test. These results can explain why in the neutron radiation test, we could not find a run where the scrubbing technique failed due to the massive accumulation of faults in the protected area.

VIII. Conclusions And Future Work

We have presented an experimental evaluation of a novel scrubbing technique. This technique presents good characteristics in terms of area and energy overhead with low repair latency compared with other solutions. Radiation experiment results and fault injection campaigns have demonstrated the effectiveness of the proposed technique to correct MBU in the configuration memory. The FLR-scrubbing technique was able to restore correctly all the frames under radiation. However, when upsets occurred in the scrubber circuit or the ICAP logic, errors were observed after the execution of the scrubbing. Future work has to focus on the improvement of the reliability of the scrubber circuit to reduce the wrong behaviors showed in the radiation results and also some possible fault tolerant techniques in the ICAP. A mechanism of self-test before scrubbing may be needed to avoid a wrong scrubbing of the configuration memory.

References

[1] V. Chandra and R. Aitken, "Impact of Technology and Voltage Scaling on the Soft Error Susceptibility in Nanoscale CMOS," in *2008 IEEE Int. Symp. on Defect and Fault Tolerance of VLSI Systems*, 2008, pp. 114–122.

[2] H. Quinn, P. Graham, J. Krone, M. Caffrey, and S. Rezgui, "Radiation-induced multi-bit upsets in SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2455–2461, Dec. 2005.

[3] H. Quinn, K. Morgan, P. Graham, J. Krone, M. Caffrey, and K. Lundgreen, "Domain Crossing Errors: Limitations on Single Device Triple-Modular Redundancy Circuits in Xilinx FPGAs," in *IEEE Trans. Nucl. Sci.*, vol. 54, no. 6, pp. 2037–2043, Dec. 2007.

[4] M. J. Wirthlin, H. Takai, and A. Harding, "Soft error rate estimations of the Kintex-7 FPGA within the ATLAS Liquid Argon (LAr) Calorimeter," *J. Instrum.*, vol. 9, no. 01, pp. C01025–C01025, Jan. 2014.

[5] I. Herrera-Alzu and M. Lopez-Vallejo, "Design Techniques for Xilinx Virtex FPGA Configuration Memory Scrubbers," in *IEEE Trans. Nucl. Sci.*, vol. 60, no. 1, pp. 376–385, Feb. 2013.

[6] J. Tonfat, F. L. Kastensmidt, R. Reis, "Energy Efficient Frame-level Redundancy Scrubbing Technique for SRAM-based FPGAs," in *2015 NASA/ESA Conf. on Adaptive Hardware and Systems (AHS)*, 2015.

[7] P. S. Ostler, M. P. Caffrey, D. S. Gibelyou, P. S. Graham, K. S. Morgan, B. H. Pratt, H. M. Quinn, and M. J. Wirthlin, "SRAM FPGA Reliability Analysis for Harsh Radiation Environments," in *IEEE Trans. Nucl. Sci.*, vol. 56, no. 6, pp. 3519–3526, Dec. 2009.

[8] M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K. a. LaBel, M. Friendlich, H. Kim, and A. Phan, "Effectiveness of Internal Versus External SEU Scrubbing Mitigation Strategies in a Xilinx FPGA: Design, Test, and Analysis," in *IEEE Trans. Nucl. Sci.*, vol. 55, no. 4, pp. 2259–2266, Aug. 2008.

[9] Xilinx Inc., Virtex-5 Configuration User Guide UG191 (v3.11), 2012.

[10] J. Heiner, N. Collins, and M. Wirthlin, "Fault Tolerant ICAP Controller for High-Reliable Internal Scrubbing," in *2008 IEEE Aerospace Conf.*, 2008.

[11] U. Legat, A. Biasizzo, and F. Novak, "SEU recovery mechanism for SRAM-Based FPGAs," in *IEEE Trans. Nucl. Sci.*, vol. 59, no. 5 PART 3, pp. 2562–2571, Oct. 2012.

[12] Xilinx Inc., LogiCORE IP Soft Error Mitigation Controller v4.0, 2013.

[13] I. Herrera-Alzu and M. López-Vallejo, "Self-reference scrubber for TMR systems based on Xilinx Virtex FPGAs," in *Lecture Notes in Computer Science*, 2011, vol. 6951 LNCS, pp. 133–142.

[14] P. M. B. Rao, M. Ebrahimi, R. Seyyedi, and M. B. Tahoori, "Protecting SRAM-based FPGAs Against Multiple Bit Upsets Using Erasure Codes," in *Proc. 51st Annu. Des. Autom. Conf. Des. Autom. Conf. - DAC'14*, pp. 1–6, 2014.

[15] C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson, and B. Hutchings, "RapidSmith: Do-It-Yourself CAD Tools for Xilinx FPGAs," in *2011 21st Int. Conf. F. Program. Log. Appl.*, pp. 349–355, Sep. 2011.

[16] A. K. Chapman, "SEU Strategies for Virtex-5 Devices," XAPP864 v2.0, 2010.

[17] E. Yang, K. Huang, Y. Hu, X. Li, J. Gong, H. Liu, and B. Liu, "HHC: Hierarchical hardware checkpointing to accelerate fault recovery for SRAM-based FPGAs," in *Proceedings of the 2013 IEEE 19th International On-Line Testing Symposium, IOLTS 2013*, 2013, pp. 193–198.

[18] M. Violante, L. Sterpone, A. Manuzzato, S. Gerardin, P. Rech, M. Bagatin, A. Paccagnella, C. Andreani, G. Gorini, A. Pietropaolo, G. Cardarilli, S. Pontarelli, and C. Frost, "A New Hardware/Software Platform and a New 1/E Neutron Source for Soft Error Studies: Testing FPGAs at the ISIS Facility," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 4, pp. 1184–1189, Aug. 2007.

[19] JEDEC, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices," Tech. Rep. JESD89A, 2006, JEDEC Standard.

[20] J. Tarrillo, J. Tonfat, L. Tambara, F. L. Kastensmidt, and R. Reis, "Multiple fault injection platform for SRAM-based FPGA based on ground-level radiation experiments," in *2015 16th Latin-American Test Symp. (LATS)*, 2015, pp. 1–6.